CSCI 361 Lecture 11: Turing Machines

Shikha Singh

Announcements & Logistics

- **HW 3** was due last night
- HW 4 released, due **Wed Oct 16** at 10 pm
- Hand in **reading questions # 8**
- No lecture on Tues (Oct 15): Reading period
  - No reading assignment for Thurs Oct 17/ Tues Oct 22
- Reminder: tomorrow colloquium (if not Mountain Day)
  - What I did Last Summer (Research)
- CSCI 361 Midterm on **Oct 22 (Tuesday)**:
  - In class exam, open notes, 75 mins
  - Will release practice exam/questions early next week

Last Time

- Intuition behind the equivalence CFL $\iff$ NPDA
- Pumping lemma for CFL and how to use it

Today

- Wrap up CFLs
- Start new model of computation: Turing machines

Pumping Lemma: CFLs

- **Statement**: If L is a CFL, then there is a number p (the pumping length) where for any $s \in L$ of length at least p, it is possible to divide s into five pieces $s = uvxyz$ satisfying the conditions

    1. $|vy| > 0$
    2. $|vxy| \leq p$
    3. For each $i \geq 0$, $uv^ixy^iz \in L$

- Note that vxy can appear anywhere in the string as long as they are no longer than p symbols long

Pumping Lemma Questions

- **Question**. What does it mean for a L to satisfy the pumping lemma?
- **Question**. What does it mean to show that L does not satisfy PL?
- **Question**. If a language satisfies PL for CFLs, does it mean it is context-free?
- **Question**. If a language is context-free, does it have to satisfy PL?

Pumping Lemma Proof Tips

- Proofs using the PL devolve to examining a bunch of cases
    - Can become painful to read/write
- Try to use closure properties whenever possible
- Try to select w that will lead to as few cases as possible
- Try to cover as many similar cases at once as possible: if several cases are analogous, address them in one general argument

CFL: Intersection Closure

- **Theorem.** If C is a context-free language and R is a regular language then $L \cap R$ is context-free.
- Proof Idea.
  - P be a PDA that recognizes C and M be DFA that recognizes R
  - Let Q, Q' be the set of states of P, M, create a new PDA P' with states $Q \times Q'$
  - P' simulates P as well as M and accepts a string if both accept
    - Ignores P's stack on M's transitions, just remembers states of M

CFL: Intersection Closure

- **Note.** Intersection of two CFLs is not necessarily context-free!
  - Example?

Context-Free or Not?

- **Question.** One of these languages is CF and the other is not, can you identify which is which?
  - $L_1 = \{w\ a^n\ b^n\ w^R \mid w \in \{a, b\}^*, n \geq 0\}$
  - $L_2 = \{w\ a^n\ w^R\ b^n \mid w \in \{a, b\}^*, n \geq 0\}$

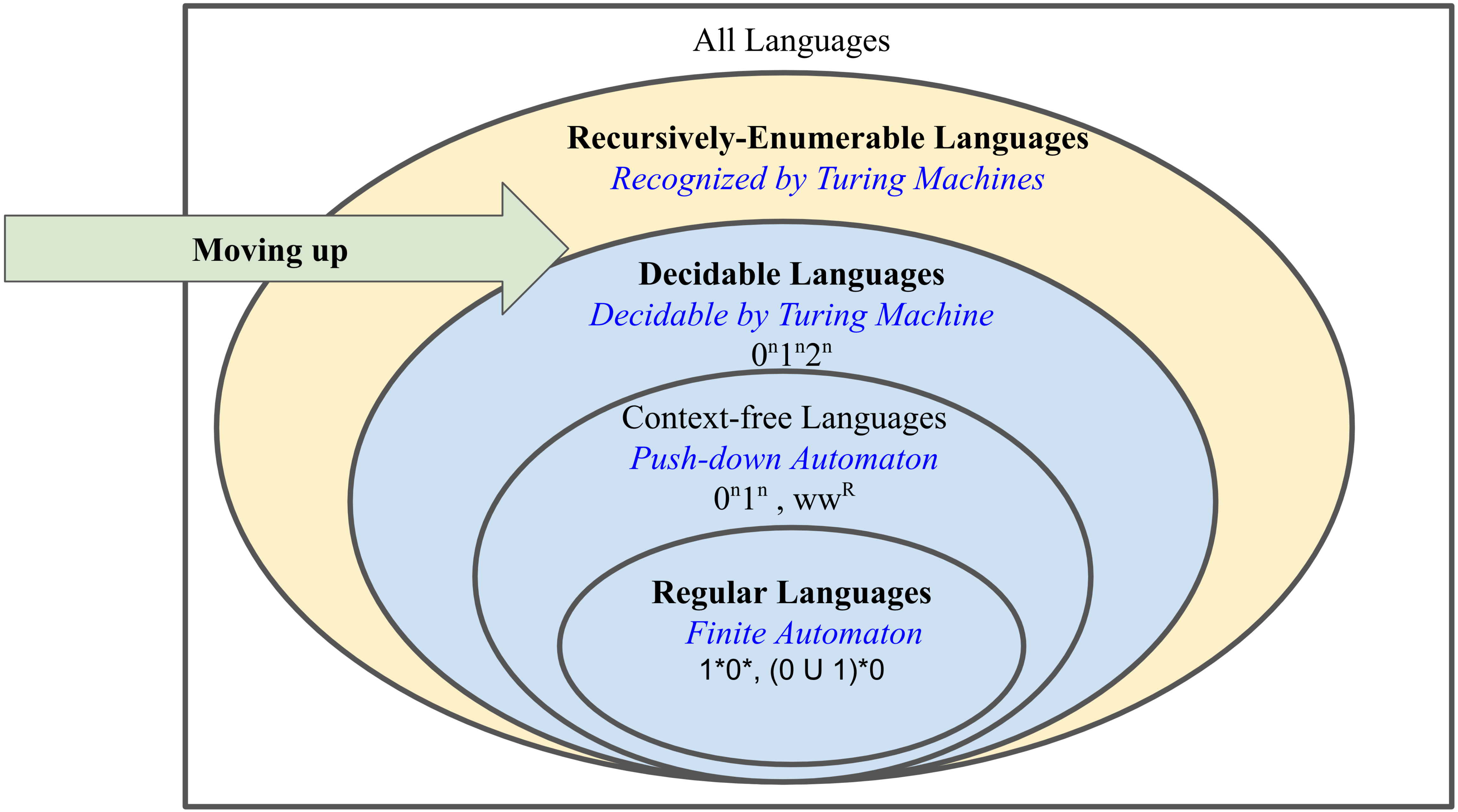
Context-Free or Not?

- $L_1 = \{w\ a^n\ b^n\ w^R \mid w \in \{a, b\}^*, n \geq 0\}$
- $L_2 = \{w\ a^n\ w^R\ b^n \mid w \in \{a, b\}^*, n \geq 0\}$
- **Answer.** L_1 is context-free but L_2 is not.
- Intuition: need to match two "pairs": can do it if they are next to each other but not if they are separated
- CFG for L_1?
  - $S \rightarrow aSa \mid bSb \mid A$
  - $A \rightarrow aAb \mid \varepsilon$
- **Exercise.** Can show L_2 is not CF using the pumping lemma, use $w = b^p a^p b^p b^p$

Examples of Non CFLs

- Pairing/Counting examples we have seen:
  - $\{a^n b^n c^n \mid n \geq 0\}$, $\{a^n b^n a^n\}$, $\{ww \mid w \in \{a,b\}^*\}$
  - HW: language of palindromes with equal # of 1s and 0s
  - Strings over $\{a, b, c\}$ with equal # of a's, b's and c's
  - $\{a^n b^m a^n b^m \mid n, m \geq 0\}$
  - $\{w\ a^n\ w^R\ b^n \mid w \in \{a,b\}^*, n \geq 0\}$
- Non-linear counting examples:
  - $\{a^{2^n} \mid n \geq 0\}$, $\{a^p \mid \text{p is a prime}\}$, $\{a^{n^2} \mid n \geq 0\}$
  - Intuition: structure is too rigid to be able to be "pumped"

Moving Up

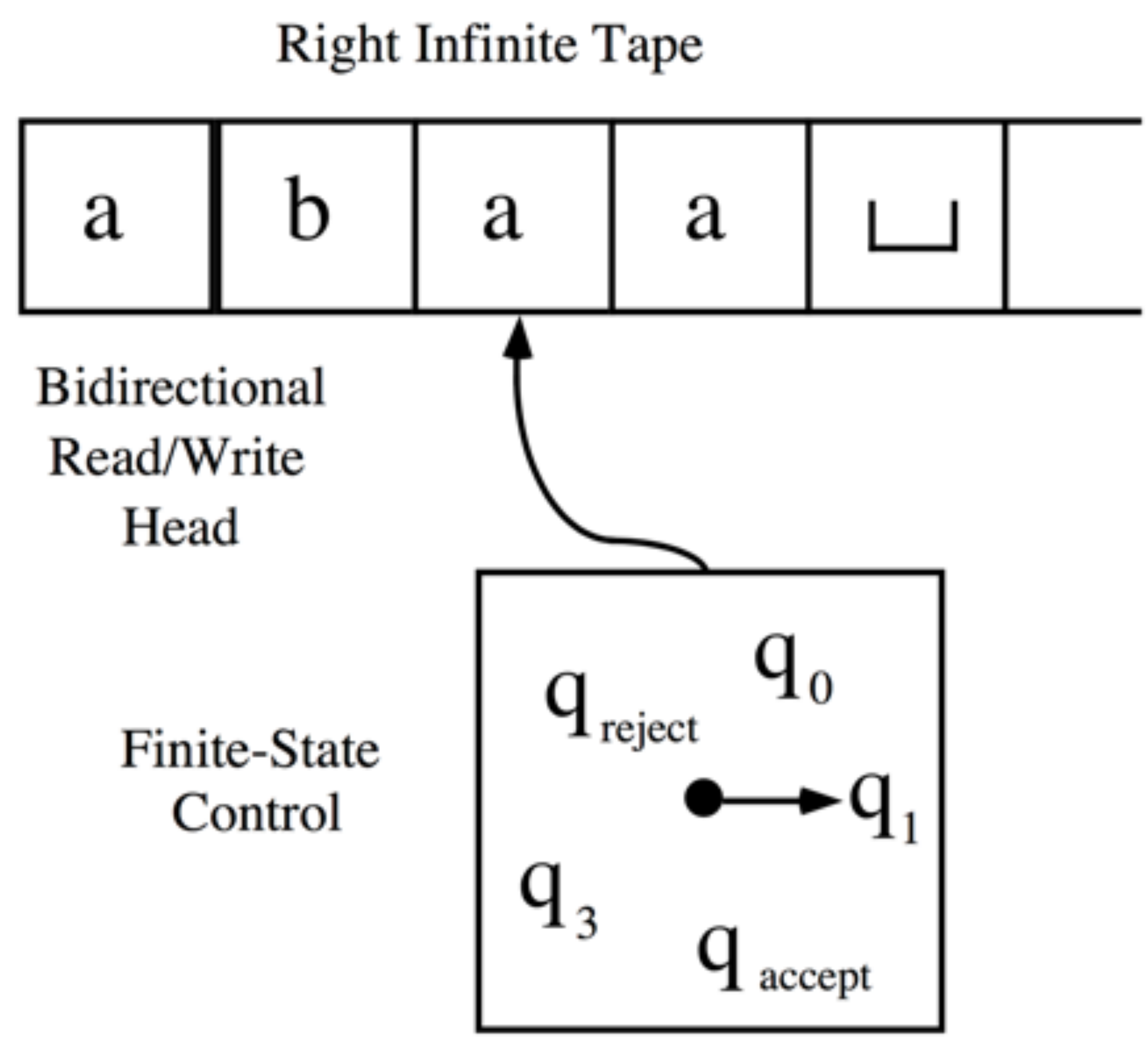

All Languages
Recursively-Enumerable Languages
Recognized by Turing Machines
Moving up
Decidable Languages
Decidable by Turing Machine
$0^n1^n2^n$
Context-free Languages
Push-down Automaton
0^n1^n , ww^R
Regular Languages
Finite Automaton
1*0*, (0 U 1)*0

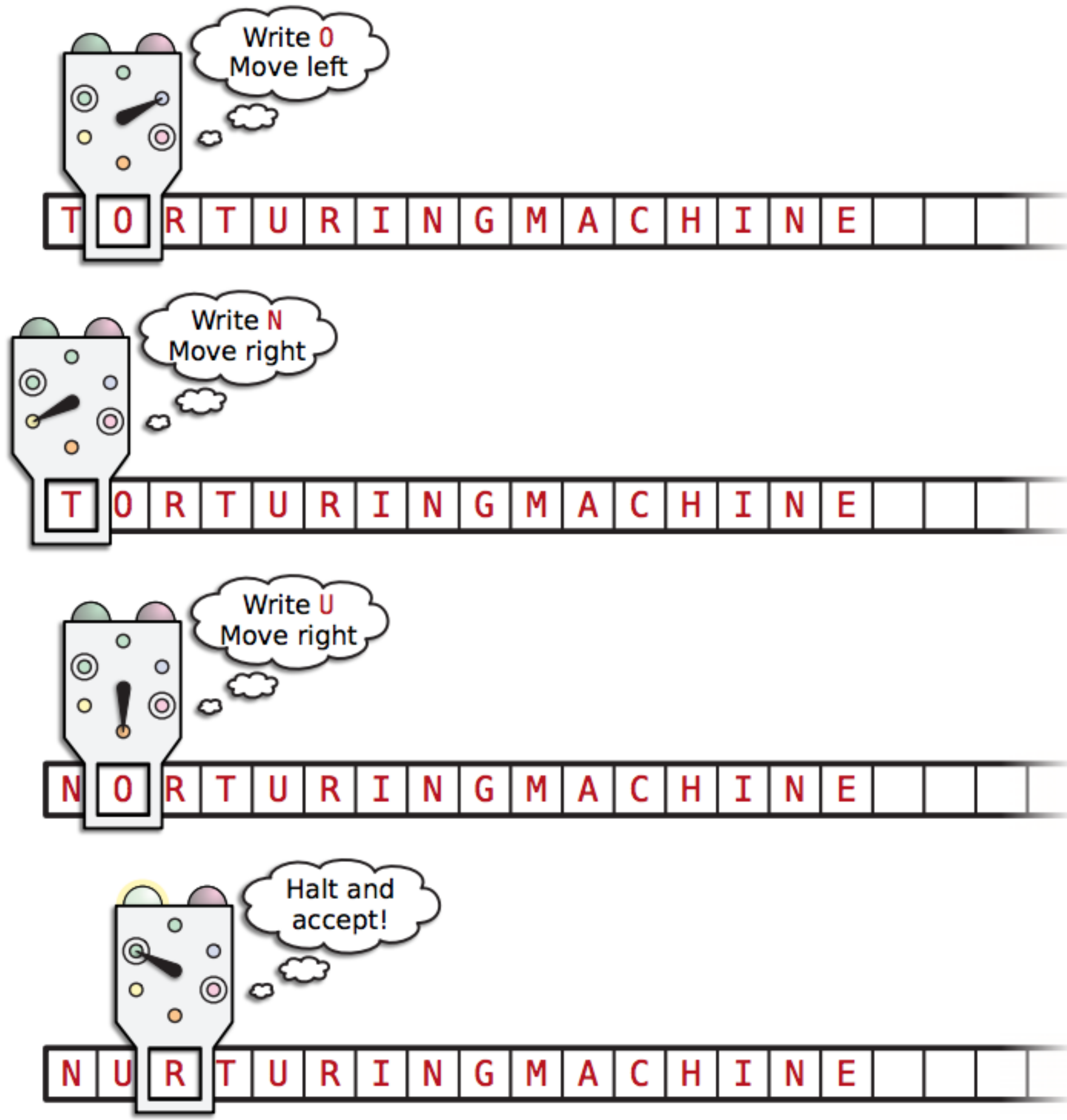
Firing Squad Problem?

- https://youtu.be/xV1aKUdlIjU?si=yM4N4WiLNYL-QKnT

Turing Machines

- Finite number of states
- Infinite tape (memory)
- Read-write head that can move right and left on the tape
- Can modify the input
- Special accept/reject states





A few iterations of a six-state Turing machine.

Image by Jeff Erickson

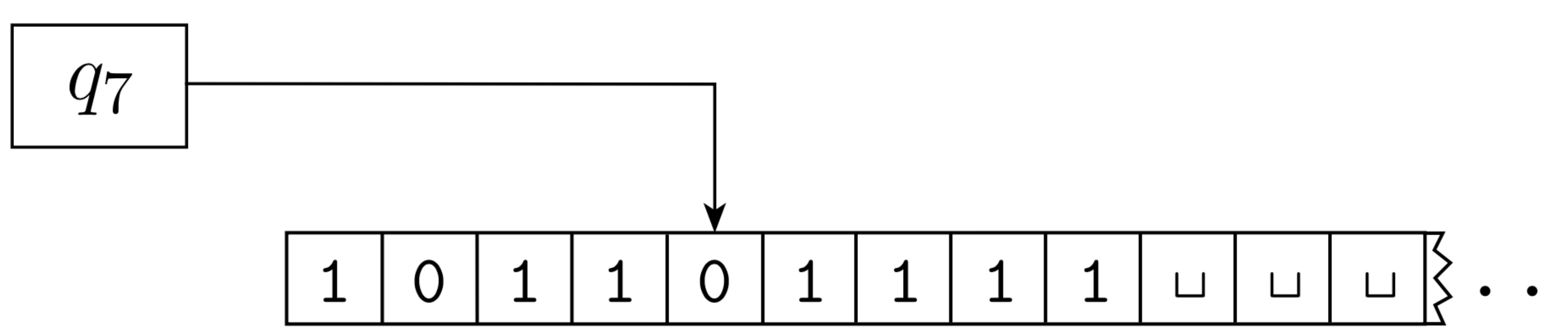
Formal Definition

- A Turing Machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets
  - Q is the set of **states**
  - Σ is the **input alphabet** and does not contain the ***blank symbol*** $\sqcup$
  - Γ is the **tape alphabet** where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the **transition function**
  - $q_0, q_{\text{accept}}, q_{\text{reject}} \in Q$ are the **start**, **accept** and **reject** states where $q_{\text{accept}} \neq q_{\text{reject}}$

How a TM Computes

- Initially, input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n squares, rest has $\sqcup$ and **head** of the TM in the leftmost position
- The computation proceeds using δ: can move left or right, alter tape contents and change states

- **Configuration** of a TM: current state, tape contents & head location
  - Written as uqv: Current state is q, current tape contents is uv, current head location is first symbol of v

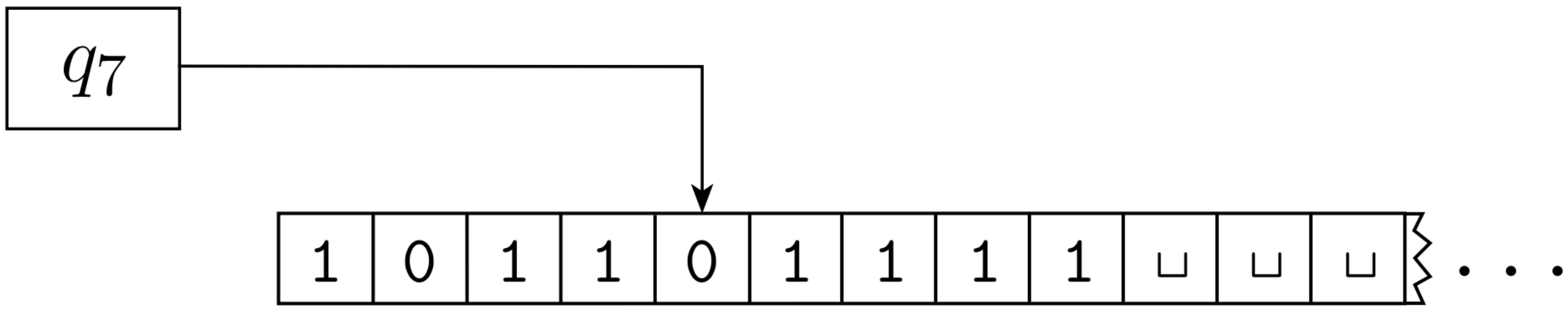


How a TM Computes

- A configuration C_1 **yields** a configuration C_2 if the TM can legally go from C_1 to C_2 using its transition function
- Consider symbols $a, b, c \in \Gamma$ and strings $u, v \in \Gamma^*$ then

$ua\ q_i\ bv$ yields $u\ q_j\ acv$ if $\delta(q_i, b) = (q_j, c, L)$, and

$ua\ q_i\ bv$ yields $uac\ q_j\ v$ if $\delta(q_i, b) = (q_j, c, R)$



Language of a TM

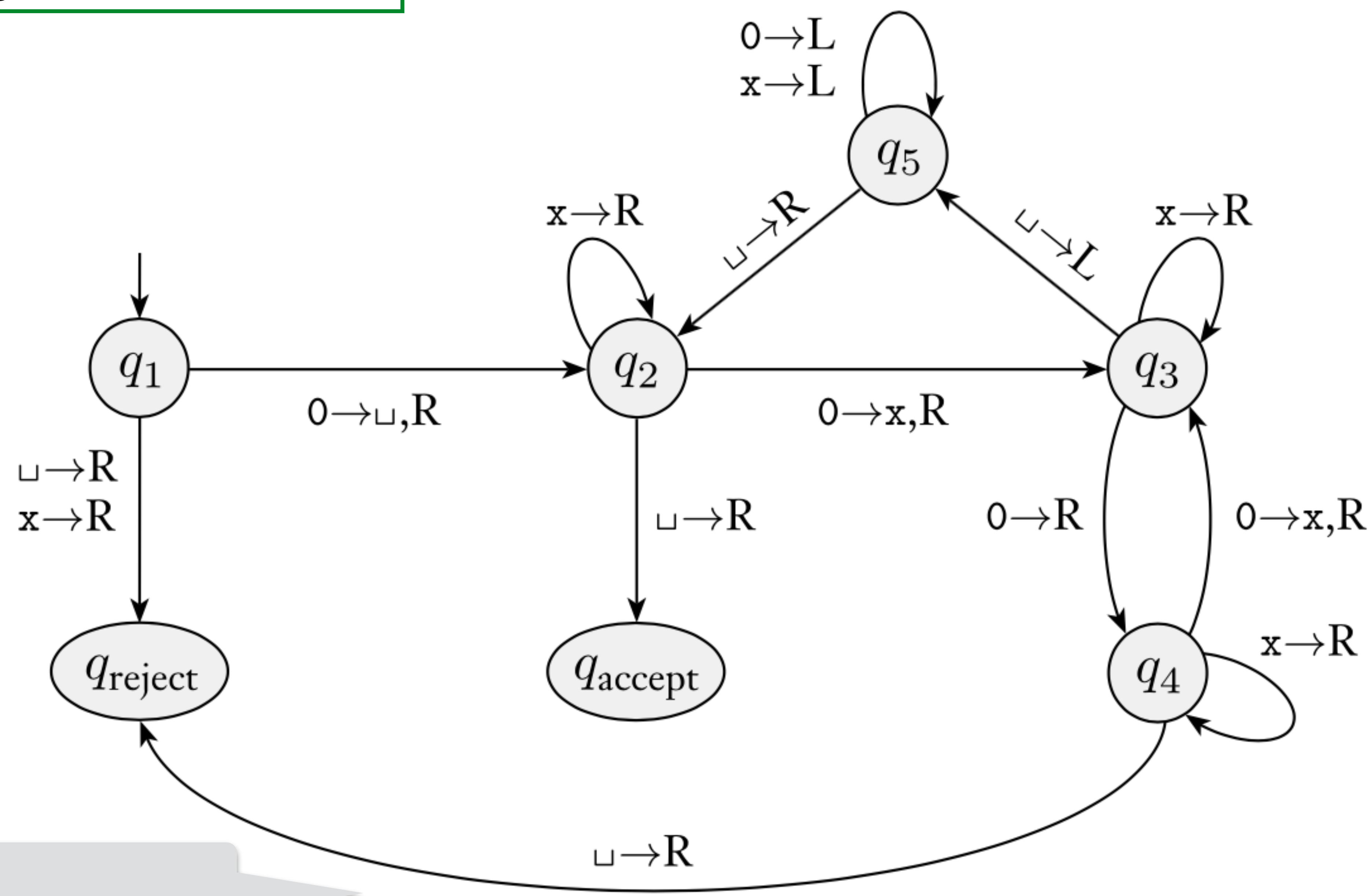
- **Start** configuration: $q_0 w$
- **Accepting** configuration if the current state is q_{accept}
- **Rejecting** configuration if the current state is q_{reject}
- A TM M ***accepts*** an input w if a sequence of configurations $C_1, \ldots, C_k$ exist such that
  - C_1 is the start configuration, each C_i yields C_{i+1} and C_k is an accepting configuration
- The set of strings accepted by M is the language ***recognized*** by M, denoted $L(M)$

Turing Machine Loops

- An important distinction between DFA/PDA and a TM
- On an input w, a TM can:
  - Accept w (and halt)
  - Reject w (and halt)
  - "Loop" on an input w (never halt): ***this is new!***
- **Definition (Decidable).** A language L is **TM-decidable** or decidable if there is a TM that accepts every string in L and rejects every string not in L (i.e., it halts on all inputs in Σ^*)
  - A TM is **decider** if it halts on every input in Σ^*

- **Example TM**: Consider a TM for the language $A = \{0^{2^n} \mid n \geq 0\}$

Each transition of the form x → y, D means "upon reading x, replace it with symbol y and move the tape head in direction D". If y is omitted x is left unchanged



State diagram: low-level description

Medium-Level Description

Consider a TM M for for the language $A = \{0^{2^n} \mid n \geq 0\}$:

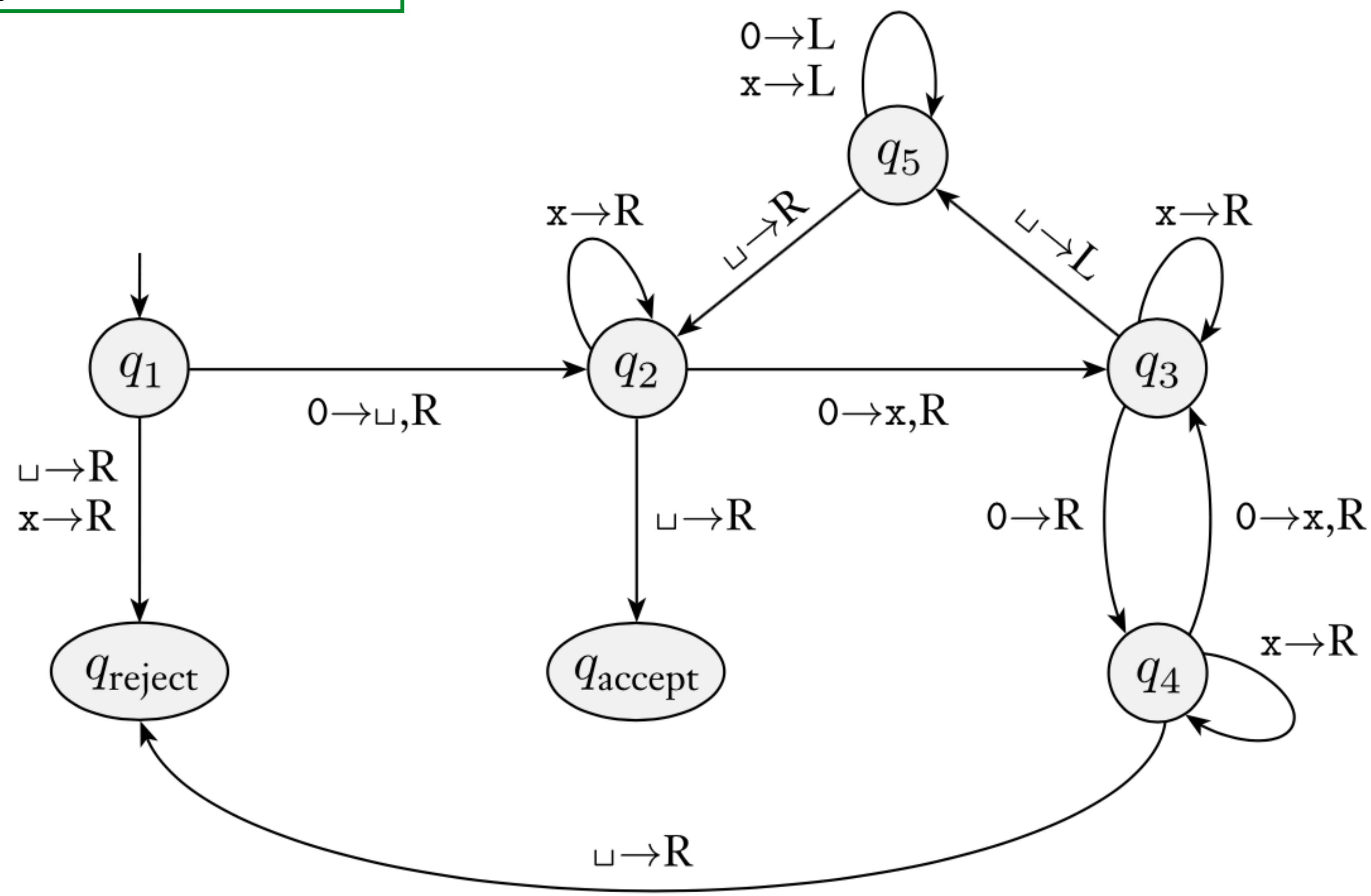
$M =$ "On input string w,

1. Sweep left to right across the tape, crossing off every other zero.
2. If in Stage 1 and there is a single zero, ***accept***
3. If in Stage 1 and there are more than one odd zeros, ***reject***
4. Return to the lefthand end of tape and go to stage 1."

Call such description medium level: says how the TM works but not as explicit as a state-diagram.

- **Example TM**: Consider a TM for the language $A = \{0^{2^n} \mid n \geq 0\}$

Each transition of the form x → y, D means "upon reading x, replace it with symbol y and move the tape head in direction D". If y is omitted x is left unchanged



Levels of Description

- Low-level description using δ and state diagram provides a complete picture but quickly become unwieldy
- Stick to "medium-level" description from now on
  - Describes how the TM works in English
  - What is OK: can include anything in a high-level description, as long as you are convinced that, if you had to, you could design a (low-level) Turing machine for it!
- We will move on to high-level descriptions (algorithms) later

Practice

- **Exercise.** Give a medium-level description of a TM that recognizes $L = \{a^n b^n c^n \mid n \geq 0\}$

Why Study Turing Machines

- Not a good model to think about ***fast*** computation
- Fast algorithms are a subject of CS 256
- In this class, we are interested in finding out if we can solve a problem at all
- To show a problem is not solvable, we need a model of what it means to solve a problem
  - Church-Turing Thesis:

Intuitive notion of algorithms	equals	*Turing machine algorithms*